

AD-A104 111 CALIFORNIA UNIV BERKELEY OPERATIONS RESEARCH CENTER F/G 12/1
MINIMIZING EXPECTED MAKESPAN IN STOCHASTIC OPEN SHOPS. (U)
JUL 81 M L PINEDO, S M ROSS AFUSR-81-0122
UNCLASSIFIED OHC-81-18 NL

END
DATE 10-81
FILED 10-81
BTIC

~~LEVEL~~

ORC 81-18
JULY 1981

AD A104111

MINIMIZING EXPECTED MAKESPAN IN STOCHASTIC OPEN SHOPS

by

MICHAEL L. PINEDO

and

SHELDON M. ROSS

(12) B5

DTIC FILE COPY

OPERATIONS
RESEARCH
CENTER

DISTRIBUTION STATEMENT A

Approved for public release:
Distribution Unlimited

UNIVERSITY OF CALIFORNIA • BERKELEY

81 9 11 056

(12)

MINIMIZING EXPECTED MAKESPAN IN STOCHASTIC OPEN SHOPS

by

Michael L. Pinedo[†]
School of Industrial and
Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia

and

Sheldon M. Ross^{††}
Department of Industrial Engineering
and Operations Research
University of California, Berkeley



JULY 1981

ORC 81-18

[†]Partially supported by the Office of Naval Research with the Center for Production and Distribution of Research, Georgia Institute of Technology.

^{††}Partially supported by the Air Force Office of Scientific Research (AFSC), USAF, under Grant AFOSR-81-0122 with the University of California. Reproduction in whole or in part is permitted for any purpose of the United States Government.

DISTRIBUTION STATEMENT A
Approved for public release; Distribution Unlimited

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

14 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
REPORT NUMBER ORC-81-18	2. GOVT ACCESSION NO. AD-A104 112	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MINIMIZING EXPECTED MAKESPAN IN STOCHASTIC OPEN SHOPS.	5. TYPE OF REPORT & PERIOD COVERED Research Report	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Michael L. Pinedo and Sheldon M. Ross	8. CONTRACT OR GRANT NUMBER(s) AFOSR-81-0122	9. PERFORMING ORGANIZATION NAME AND ADDRESS Operations Research Center University of California Berkeley, California 94720
11. CONTROLLING OFFICE NAME AND ADDRESS United States Air Force Air Force Office of Scientific Research Bolling Air Force Base, D.C. 20332	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2304/A5	12. REPORT DATE July 1981
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	13. NUMBER OF PAGES 24	15. SECURITY CLASS. (of this report) Unclassified
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.	17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Open Shop Makespan Two Machines		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) (SEE ABSTRACT)		

ABSTRACT

Suppose that two machines are available to process n tasks. Each task has to be processed on both machines, the order in which this happens is immaterial. Task j has to be processed on machine 1 (2) for random time $X_j(Y_j)$ with distribution $F_j(G_j)$. This kind of model is usually called an Open Shop. The time that it takes to process all tasks is normally called the makespan. Every time a machine finishes processing a task the decision-maker has to decide which task to process next.

Assuming that X_j and Y_j have the same exponential distribution we show that the optimal policy instructs the decision-maker, whenever a machine is freed, to start processing the task with longest expected processing time among the tasks still to be processed on both machines. If all tasks have been processed at least once, it does not matter what the decision-maker does, as long as he keeps the machine busy. We then consider the case of n identical tasks and two machines with different speeds. The time it takes machine 1 (2) to process a task has distribution $F(G)$. Both distributions F and G are assumed to be New Better than Used (NBU) and we show that the decision-maker stochastically minimizes the makespan when he always gives priority to those tasks which have not yet received processing on either machine.

Accession No.	MS. A. 1.61
Date Rec'd	1983
Comments	
Classification	
By	
Distribution	
Availability	1996
Acq. Agent	John C. Miller
Notes	

MINIMIZING EXPECTED MAKESPAN IN STOCHASTIC OPEN SHOPS

by

Michael L. Pinedo and Sheldon M. Ross

1. INTRODUCTION AND SUMMARY

Suppose that two machines are available to process n tasks. Each task has to be processed on both machines, the order in which this happens is immaterial. Task j has to be processed on machine 1 (2) for random time $X_j (Y_j)$ with distribution $F_j (G_j)$. This kind of model is usually called an Open Shop. The time that it takes to process all tasks is normally called the makespan. Every time a machine finishes processing a task the decision-maker has to decide which task to process next. A policy prescribes, as a function of the state of the system, the actions to take at the various decision moments; of interest are the policies that minimize the expected makespan.

Gonzalez and Sahni [1] considered the case where the processing times are deterministic and presented an $O(n)$ algorithm to find an optimal schedule. The case of stochastic processing times has, to our knowledge, not yet been considered in the literature.

In the first section of this paper, we assume that X_j and Y_j have the same exponential distribution with rate μ_j . We show that the optimal policy instructs the decision-maker, whenever a machine is freed, to start processing the task with longest expected processing time among the tasks still to be processed on both machines. If all tasks have been processed at least once, it does not matter what the decision-maker does, as long as he keeps the machine busy.

In the second section we consider the case of n identical tasks and two machines with different speeds. The time it takes machine 1 (2) to process a task has distribution $F(G)$. Both distributions F and G are New Better than Used (NBU). We show that the decision-maker stochastically minimizes the makespan when he always gives priority to those tasks which have not yet received processing on either machine.

In the last section the case of n identical tasks and two identical machines will be considered. The processing time of a task on a machine has an exponential distribution with rate one. A closed form expression for the expected makespan under the optimal policy is given.

2. THE CASE OF EXPONENTIAL PROCESSING TIMES

In this section we assume that X_j and Y_j have the same exponential distribution with rate μ_j .

Note that a policy only has to instruct the decision-maker what to do as long as there are still tasks waiting for their first processing. This is true for the following reason: when machine 1 (2) becomes free, the decision-maker can otherwise choose only from tasks which have to be processed on machine 1 (2) alone and the sequence in which these tasks will be processed on machine 1 (2) will not affect the makespan.

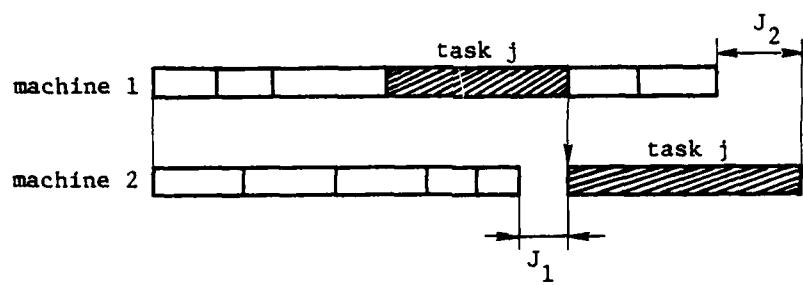
If M denotes the makespan then clearly

$$(1) \quad M \geq \max \left(\sum_{i=1}^n X_i, \sum_{i=1}^n Y_i \right).$$

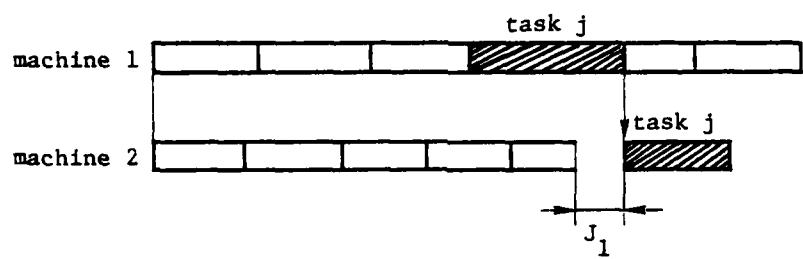
The makespan may be strictly larger than the R.H.S. of (1) when one machine is kept idle for some time in between the processing of two tasks. A distinction will be made between two types of idle periods (see Figure 1).

In an idle period of type II, a machine is kept idle for some time, say J_1 , then processes the last task, say task j , and finishes processing task j while the other machine is still busy. It is clear that, although a machine has been kept idle for some time, the makespan is nevertheless equal to the R.H.S. of (1). In an idle period of type I, a machine is kept idle for some time, say J_1 , then processes the last task, say task j , and finishes processing this task some time, say J_2 , after the other machine has finished processing its tasks. Now

$$(2) \quad M = \max \left(\sum_{i=1}^n X_i, \sum_{i=1}^n Y_i \right) + N$$



Idle Period of Type I



Idle Period of Type II

FIGURE 1

where

$$N = \min (J_1, J_2) .$$

It can be verified easily that only one task can cause an idle period and an idle period has to be either of type I or type II. In the remaining part of the paper superscripts will consistently refer to policies, e.g. M^0 denotes the makespan under policy π^0 , N^0 denotes the second term on the R.H.S. of (2) under π^0 , etc. Let p_j^0 denote the probability that task j causes an idle period of type I under policy π^0 . Then, after taking expectations on both sides of (2), we obtain

$$(3) \quad E(M^0) = E\left(\max\left(\sum_{i=1}^n X_i, \sum_{i=1}^n Y_i\right)\right) + E(N^0)$$

where

$$E(N^0) = \sum_{j=1}^n p_j^0 \cdot \frac{1}{2\mu_j} .$$

As the first term on the R.H.S. of (3) does not depend on the policy, it suffices to find a policy which minimizes the second term on the R.H.S. of (3).

Before continuing with our analysis of the Open Shop model we will consider a simpler scheduling model; a model where the tasks have to be processed only once on either one of the two machines. This model has been treated in the literature extensively for various objective functions, see [2], [3], [4]. Assume that at decision moment t one machine is free and the other one is busy processing task 0. A policy j_1, j_2, \dots, j_r , a permutation of $1, 2, \dots, r$, prescribes that at time t task j_1 is

put on the empty processor; the next time a processor is freed task j_2 is started, etc. For this model we have

Lemma 1:

Let q_j^1 (q_j^2), $j = 0, 1, 2, \dots, r$, denote the probability that the last task to be finished under policy $\pi^1 = 1, 2, 3, \dots, r-1, r$ ($\pi^2 = 2, 1, 3, \dots, r-1, r$) is task j . When $u_1 = \min(u_1, u_2, \dots, u_r)$, then, for any task 0, not necessarily exponentially distributed,

$$\begin{aligned} q_0^1 &= q_0^2 \\ q_1^1 &\leq q_1^2 \\ q_j^1 &\geq q_j^2 \quad j = 2, \dots, r \end{aligned}$$

and

$$\sum_{i=1}^r q_i^1 = \sum_{i=1}^r q_i^2.$$

Proof:

See Lemma 1 in Pinedo and Weiss [2]. ■

Although Lemma 1 was originally set up to analyze a simpler model, where tasks have to be processed only once on either one of the two machines, it will play an important role in the analysis of the more complicated Open Shop model we have under consideration in this paper. In Theorem 1 the lemma will be used repeatedly, applied on a set of tasks which after a specific time t have to undergo their first processing on either one of the two machines. As long as there are tasks waiting for their first processing no task will start its second processing.

We will be interested in the probabilities of each task of this set being the last one of the set to finish its first processing. These probabilities are needed when computing the probability of a task causing an idle period of type I, as for a task to cause a type I idle period it must have been the last task to finish its first processing.

Theorem 1:

The policy that minimizes the expected makespan is the policy which, whenever either of the two machines is free, instructs the decision-maker

- (i) when there are still tasks which have not yet received processing on either machine, to start among these the one with the largest expected processing time and
- (ii) when all tasks have been processed at least once, to start any one of the tasks still to be processed on the machine just freed.

Proof:

Assume at decision moment t one processor is idle and the second one is processing task 0 with an arbitrary distribution. Let π^* denote the policy stated in the theorem. In order to prove that π^* is optimal it suffices to show that using π^* from t onwards results in a smaller expected makespan than acting differently at t and using π^* from the next decision-moment onwards. Two types of actions at t which are not according to π^* have to be considered. Firstly, it is possible to start a task which has not yet been processed on either machine, but which is not the largest task among the tasks not been processed on either machine. Secondly, it is possible to start a task which already has been processed on the other machine.

The following notation will be used: $\{A\}$ denotes the set of tasks that at t did not yet finish their first processing, while $\{B\}$ denotes the set of tasks that at t did not yet start their second processing. Clearly $\{B\} \supset \{A\}$.

Case 1:

Let π^1 denote the policy which instructs the decision-maker at time t to start task m with rate μ_m , $m \in A$, where m is not the largest task among tasks in $\{A\}$ and to adopt policy π^* from the next decision moment onwards. Let q_j^1 (q_j^*), $j \in A$, denote the probability that task j is the last task to finish its first processing under π^1 (π^*) and therefore be a candidate to cause an idle period. Suppose this task j would be processed on machine 1--for it to cause a type I idle period it also has to outlast all those tasks which still have to receive their second processing on machine 2, and then, after task j finishes on machine 1 and starts on machine 2, it has to outlast those tasks which have yet to receive their second processing on machine 1. So

$$p_j^* = q_j^* \cdot \prod_{i \in \{B-j\}} \frac{\mu_i}{\mu_i + \mu_j}$$

and

$$p_j^1 = q_j^1 \cdot \prod_{i \in \{B-j\}} \frac{\mu_i}{\mu_i + \mu_j}.$$

Also

$$p_0^1 = p_0^*.$$

Observe that p_j^* and p_j^1 do not depend on which machine task i ($i \neq j$) has to be processed the second time. If task i would be processed the second time on the same machine it was processed the first time, it would affect neither p_j^* nor p_j^1 . This fact will be used in Case 2. In order to show that $E(N^*) \leq E(N')$ we have to show that

$$(4) \sum_{j \in \{A-0\}} \left(\frac{1}{2u_j} \cdot q_j^* \cdot \prod_{i \in \{B-j\}} \frac{u_i}{u_i + u_j} \right) \leq \sum_{j \in \{A-0\}} \left(\frac{1}{2u_j} \cdot q_j^1 \cdot \prod_{i \in \{B-j\}} \frac{u_i}{u_i + u_j} \right).$$

As a prelude to showing the above, note that when $u_k \leq u_\ell$

$$(5) \frac{1}{2u_k} \cdot \prod_{i \in \{B-k\}} \frac{u_i}{u_i + u_k} \geq \frac{1}{2u_\ell} \cdot \prod_{i \in \{B-\ell\}} \frac{u_i}{u_i + u_\ell}.$$

Suppose the sequence in which tasks $\{A\}$ start their first processing under π' is $m, 1, 2, \dots, m-1, m+1, \dots, r$ where $u_1 \leq u_2 \leq \dots \leq u_{m-1} \leq u_m \leq u_{m+1} \leq \dots \leq u_r$. Performing a pairwise switch in this sequence gives $1, m, 2, \dots, m-1, m+1, \dots, r$. Let this new sequence correspond to policy π'' . Using the relationship between q_j' and q_j'' , $j = 1, \dots, n$, (established by Lemma 1) and (5) we find that $E(N'') \leq E(N')$. After performing a series of similar adjacent pairwise interchanges, each one involving task m and the task immediately following it, (4) follows.

Case 2:

Let π''' denote the policy which instructs the decision-maker at time t to start task h with rate u_h , $h \in \{B - A\}$, and to adopt policy π^* from the next decision moment onwards. Let q_j''' , $j \in A$, denote the probability that task j is the last one of tasks in $\{A\}$ to finish its first processing and also finishes after task h finishes under π''' .

Let q_h''' denote the probability that task h finishes after all tasks of $\{A\}$ finish their first processing under π''' . Assume in the remaining part of the proof that when using π^* from t onwards we may, after having started all tasks of $\{A\}$, start task h on the machine that comes free first (we have seen under Case 1 that the probability of task j causing a type I idle period does not depend on which machine processes task h for the second time). Let q_j^* now denote the probability that task j is the last one to finish its first processing among tasks $\{A\}$ and also finishes after task h finishes. Let q_h^* denote the probability that task h finishes after all tasks of $\{A\}$ have finished their first processing. So now

$$p_j^* = q_j^* \cdot \prod_{i \in \{B-j-h\}} \frac{u_i}{u_i + u_j} \quad j \in A$$

and

$$p_j''' = q_j''' \cdot \prod_{i \in \{B-j-h\}} \frac{u_i}{u_i + u_j} \quad j \in A$$

Again

$$p_0''' = p_0^* \cdot$$

In order to show that $E(N^*) \leq E(N''')$ we have to show that

$$\sum_{j \in \{A-0\}} \left(\frac{1}{2u_j} \cdot q_j^* \cdot \prod_{i \in \{B-j-h\}} \frac{u_i}{u_i + u_j} \right) \leq \sum_{j \in \{A-0\}} \left(\frac{1}{2u_j} \cdot q_j''' \cdot \prod_{i \in \{B-j-h\}} \frac{u_i}{u_i + u_j} \right).$$

From Lemma 1 follows that $q_h^* \geq q_h'''$ and $q_i^* \leq q_i'''$, $i \in A$. From (5) follows then that $E(N^*) \leq E(N''')$. ■

3. IDENTICAL TASKS ON TWO MACHINES WITH DIFFERENT SPEEDS

In this section we assume that X_j and Y_j , $j = 1, \dots, n$, have distributions F and G respectively. Both distributions F and G are New Better than Used (NBU), i.e.,

$$\bar{F}(x+y)/\bar{F}(x) \leq \bar{F}(y) \quad x \geq 0, y \geq 0$$

and

$$\bar{G}(x+y)/\bar{G}(x) \leq \bar{G}(y) \quad x \geq 0, y \geq 0.$$

Consider decision moment t when one machine has become idle and the other is busy processing a task. Let n_{12} denote the number of tasks that have not yet received processing on either machine and let $n_1(n_2)$ denote the number of tasks that are waiting for processing only on machine 1 (2). When the task that is being processed at t on the busy machine, say machine 1 (2), is not required to be processed afterwards on the other machine, it will be called a task of type 1 (2); if after having finished its current processing on machine 1 (2), it is required to be processed afterwards on machine 2 (1) it will be called a task of type 12 (21). The type of this task will be designated by τ . Let z denote the amount of processing this task has already received at time t on machine 1 (2). The state S of the process at time t is then completely specified by the quintuple $(n_{12}, n_1, n_2, \tau, z)$. The random variable $M^0(S)$ will denote the remaining time to finish all tasks, starting in state S and using policy π^0 . In the subsequent lemma and theorem, the remaining time to finish all tasks, departing from two different states, say S_1 and S_2 , while using the same policy or using two different policies, say π^1 and π^2 , while departing from the same state will be

compared repeatedly. However, whenever $M^1(S_1)$ will be compared with $M^1(S_2)$ the amount of processing already done on the task in the busy machine will be identical in states S_1 and S_2 (the types of these tasks may however differ).

The following notation will be used: $A \supseteq B$ denotes that the random variable A is stochastically larger than random variable B , i.e.,

$$P(A \geq x) \geq P(B \geq x) \quad \forall x.$$

Lemma 2:

Under the optimal policy π^*

$$M^*(k+1, l-1, n-1, \tau, z) \supseteq M^*(k, l, m, \tau, z).$$

Proof:

The distributions of the amount of processing still to be done on either machine are the same in the two states. In state $(k+1, l-1, m-1, \tau, z)$ there is an additional constraint in the form of an extra task which has to be processed on both machines. So starting from state (k, l, m, τ, z) the decision-maker can do at least as well as when starting from state $(k+1, l-1, m-1, \tau, z)$ if he acts optimally in the two cases. ■

In the next theorem we will use repeatedly the following approach when comparing $M^1(S_1)$ with $M^1(S_2)$ or $M^1(S_1)$ with $M^2(S_1)$. The time the busy machine needs to finish the task started before t will have the same distribution in the two states which are being compared. We condition them on which machine finishes first and compare $M^1(S_1)$ with $M^1(S_2)$ (or with $M^2(S_1)$) under the condition that a particular machine finishes first.

We are now ready for the main result of this section.

Theorem 2:

The makespan is stochastically minimized if the decision-maker whenever a machine is freed starts, when possible, a task which has not yet been processed on either machine.

Proof:

Let π^* denote the policy which always gives priority to the tasks which have not yet been processed on either machine. It suffices to show that at any decision moment in any state it is better to use policy π^* than to take an action not prescribed by π^* (i.e., starting a task of either type 1 or type 2 depending upon which machine is idle) and use π^* from the next decision moment onwards. Call this last policy π' .

So we have to show that for any S

$$M^*(S) \subseteq M'(S).$$

In all inequalities to be shown the time the busy machine has been processing its task (z) is the same in the two states being compared; a state will therefore be denoted by (k, l, m, τ) .

We will show by induction on k

$$A(k) : M^*(k-1, l, m, 12) \subseteq M^*(k, l-1, m, 1)$$

$$M^*(k-1, l, m, 21) \subseteq M^*(k, l, m-1, 2)$$

$$B(k) : M^*(k, l, m, 1) \subseteq M'(k, l, m, 1)$$

$$M^*(k, l, m, 2) \subseteq M'(k, l, m, 2)$$

$$C(k) : M^*(k-1, l+1, m, 12) \subseteq M^*(k, l, m, 1)$$

$$M^*(k-1, l, m+1, 21) \subseteq M^*(k, l, m, 2)$$

$$D(k) : M^*(k, \ell, m, 12) \subseteq M'(k, \ell, m, 12)$$

$$M^*(k, \ell, m, 21) \subseteq M'(k, \ell, m, 21).$$

Showing $B(k)$ and $D(k)$ for every k, ℓ and m proves the theorem.
 At each step only the first of the two inequalities involved will be shown as the proof of the second is identical.

Induction Base: $k = 1$

A(1):

On machine 1 a task of type 12 has been receiving service for time z .
 Let Z denote the random time needed to finish this task on machine 1.
 As F is NBU

$$Z \subseteq X_i.$$

It can be established easily that

$$M^*(0, \ell, m, 12) = \max \left(\sum_{i=1}^{\ell} X_i + Z, \sum_{i=1}^{m+1} Y_i, Z + Y_1 \right)$$

where the first (second) term between the parentheses on the R.H.S. represents the total amount of work still to be done on machine 1 (2) and the third term represents the time needed to finish the task of type 12.
 The third term is the largest iff the task of type 12 causes an idle period of type I. On the other hand

$$M^*(1, \ell - 1, m, 1) = \max \left(\sum_{i=1}^{\ell} X_i + Z, \sum_{i=1}^{m+1} Y_i, X_1 + Y_1 \right)$$

so A(1) clearly holds.

B(1):

It was shown before that

$$M^*(1, \ell, m, 1) = \max \left(\sum_{i=1}^{\ell+1} x_i + z, \sum_{i=1}^{m+1} y_i, x_1 + y_1 \right).$$

Under policy π' the task which has to be processed on both machines will not be started immediately. But starting this task later is equivalent with a larger third term on the R.H.S. Clearly B(1) holds.

C(1):

This case is identical to A(1) .

D(1):

It can be shown easily that

$$M^*(1, \ell, m, 12) = \max \left(\sum_{i=1}^{\ell+1} x_i + z, \sum_{i=1}^{m+2} y_i, x_1 + y_1, z + y_2 \right)$$

where the third and fourth term represent the times needed to finish the two tasks which still have to receive processing on both machines. Under π' the only remaining task waiting for both machines will not be started immediately. But starting this task later is equivalent with a larger third term on the R.H.S. So D(1) holds.

General k :

To prove A(k), ..., D(k) for arbitrary k , assume A(k - 1), ..., D(k - 1) to hold. We will show A(k), ..., D(k) by induction on $\ell + m$.

Induction Base $\ell + m = 1$:

A(k) :

To show

$$M^*(k-1,1,0,12) \subseteq M^*(k,0,0,1) .$$

If machine 1 finishes first it suffices to show

$$M^*(k-2,1,1,21) \subseteq M^*(k-1,0,0,21) .$$

This holds because of Lemma 2. If machine 1 finishes first it suffices

to show

$$M^*(k-2,2,0,21) \subseteq M^*(k-1,1,0,1) .$$

This holds because of A(k-1) .

B(k) :

To show

$$M^*(k,1,0,1) \subseteq M'(k,1,0,1)$$

and

$$M^*(k,0,1,1) \subseteq M'(k,0,1,1) .$$

In state $(k,1,0,1)$ both π^* and π' have to take the same action so it suffices to consider state $(k,0,1,1)$. The case of machine 1 finishing first holds because of A(k) for $\ell + m = 1$. In case machine 2 finishes first the next state will be under both π^* and π' $(k-1,1,1,1)$.

C(k) :

To show

$$(i) \quad M^*(k-1,1,1,12) \subseteq M^*(k,0,1,1) .$$

The case of machine 1 finishing first holds because of Lemma 2. The case of machine 2 finishing first holds because of $C(k-1)$.

$$(ii) \quad M^*(k-1,2,0,12) \subseteq M^*(k,1,0,1) .$$

The case of machine 1 finishing first holds because of Lemma 2 and the case of machine 2 coming free first holds because of $C(k-1)$.

D(k) :

To show

$$M^*(k,1,0,12) \subseteq M'(k,1,0,12)$$

and

$$M^*(k,0,1,12) \subseteq M'(k,0,1,12) .$$

As in state $(k,1,0,12)$ both π^* and π' have to take the same action, it suffices to show only the second inequality. The case of machine 1 coming free first holds because of $C(k)$ for $l+m=1$. The case of machine 2 coming free first holds because of Lemma 2.

General $l+m$:

Assume $A(k), \dots, D(k)$ to hold for $l+m-1$.

A(k) :

To show

$$M^*(k-1, l, m, 12) \subseteq M^*(k, l-1, m, 1) .$$

The case of machine 1 coming free first can again be handled through

Lemma 2. The case of machine 2 finishing first holds because of $A(k-1)$.

B(k) :

To show

$$M^*(k, l, m, 1) \subseteq M'(k, l, m, 1) .$$

The case of machine 1 coming free first holds because of $A(k)$. The

case of machine 2 finishing first holds because of Lemma 2.

C(k) :

To show

$$M^*(k-1, l+1, m, 12) \subseteq M^*(k, l, m, 1) .$$

The case of machine 1 coming free first holds because of Lemma 2. The

case of machine 2 finishing first holds because of $C(k-1)$.

D(k) :

To show

$$M^*(k, l, m, 12) \subseteq M'(k, l, m, 2) .$$

The case of machine 1 coming free first holds because of $C(k)$. The

case of machine 2 finishing first holds because of Lemma 2. ■

4. IDENTICAL TASKS WITH EXPONENTIAL PROCESSING TIMES ON IDENTICAL MACHINES

In this section we consider the case where both X_i and Y_i ,
 $i = 1, \dots, n$, are exponentially distributed with mean one. We present
a closed form solution for the expected makespan under the optimal policy.
From Section 1 we have

$$(6) \quad E(M^*) = E\left[\max\left(\sum_{i=1}^n X_i, \sum_{i=1}^n Y_i\right)\right] + \sum_{j=1}^n p_j^* \frac{1}{2}.$$

Consider the first term on the R.H.S. of (6). Clearly

$$(7) \quad E\left[\max\left(\sum_{i=1}^n X_i, \sum_{i=1}^n Y_i\right)\right] = 2n - E\left[\min\left(\sum_{i=1}^n X_i, \sum_{i=1}^n Y_i\right)\right].$$

In order to compute the second term on the R.H.S. of (7) suppose we have
a stockpile of n type 1 and n type 2 components, each of which function
for an exponential time with mean 1. Suppose further that for the "system"
to work there must always be both a single type 1 and type 2 in service.
If T represents the system lifetime then $E[T]$ is the desired expectation.
Letting N be the number of components that fail we obtain

$$E[T] = \frac{E[N]}{2} = \frac{1}{2} \sum_{k=n}^{2n-1} k 2 \binom{k-1}{n-1} \left(\frac{1}{2}\right)^k$$

or

$$E\left[\min\left(\sum_{i=1}^n X_i, \sum_{i=1}^n Y_i\right)\right] = \sum_{k=n}^{2n-1} k \binom{k-1}{n-1} \left(\frac{1}{2}\right)^k.$$

To compute $\sum_{j=1}^n p_j^*$ assume, without loss of generality, that the ordering $1, 2, \dots, n$ is used (the ordering, of course, being irrelevant since all rates are equal). Now $p_j^* = \left(\frac{1}{2}\right)^{2n-2}$ for $j = 1$ or 2 for in either case $n-1$ services must finish before j and when j then switches machines once again $n-1$ services must finish before it. For $j > 2$ suppose at the moment the j^{th} element is first serviced by a machine there have been a total of i completions on that machine and $j-2-i$ on the other. The probability that j would cause a type 1 idle period in this case would thus be

$$\left(\frac{1}{2}\right)^{n-1-(j-2-i)} \left(\frac{1}{2}\right)^{n-1-i} = \left(\frac{1}{2}\right)^{2n-j}.$$

As the above does not depend on i we see that

$$p_1^* = \left(\frac{1}{2}\right)^{2n-2}, \quad p_j^* = \left(\frac{1}{2}\right)^{2n-j}, \quad j \geq 2$$

and so

$$\frac{1}{2} \sum_1^n p_j^* = \left(\frac{1}{2}\right)^{2n-1} + \left(\frac{1}{2}\right)^n - \left(\frac{1}{2}\right)^{2n-1} = \left(\frac{1}{2}\right)^n$$

implying that

$$E[M^*] = 2n - \sum_{k=n}^{2n-1} k \binom{k-1}{n-1} \left(\frac{1}{2}\right)^k + \left(\frac{1}{2}\right)^n.$$

REFERENCES

- [1] Gonzalez, T. and S. Sahni, "Open Shop Scheduling to Minimize Finish Time," Journal of Computing Machinery, Vol. 23, pp. 165-679 (1976).
- [2] Pinedo, M. and G. Weiss, "Scheduling Stochastic Tasks on Two Parallel Processors," Naval Research Logistic Quarterly, Vol. 26, pp. 527-535 (1979).
- [3] Weber, R., "Scheduling Jobs with Stochastic Processing Requirements on Parallel Machines to Minimize Makespan or Flow Time," to appear in Journal of Applied Probability (1981).
- [4] Weiss, G. and M. Pinedo, "Scheduling Tasks with Exponential Service Times on Nonidentical Processors to Minimize Various Cost Functions," Journal of Applied Probability, Vol. 17, pp. 187-202 (1980).

END
DATE
FILMED
10-81
DTIC